Integrity Protection and Access Control

Who do you trust?

Glenn Wurster

Principal Security Researcher

Caveat

Opinions are my own.

Physical Access for an Attacker

- With physical access to your flash file-system, the attacker can:
  - Read and modify the POSIX permissions
  - Read and modify the SELinux label
  - Read and modify the file contents
  - Read and modify directory structure
  - Do whatever they want

Who cares?

Who Does Care?

Android

(Trusted System, Untrusted Data)

Encrypt it all!

dm-crypt to the rescue†‡

† with integrity protection
‡ does not protect against root persistence

What happens if we don’t use dm-crypt?

Bad things can happen

SEAndroid Danger

file_contexts

```
/data/security(/.*)?        u:object_r:security_file:s0
```

domain.te

```
# Only system_server can create under /data/security
neverallow { domain -system_server } security_file:dir
       { rename write add_name remove_name rmdir };
```

untrusted_app.te

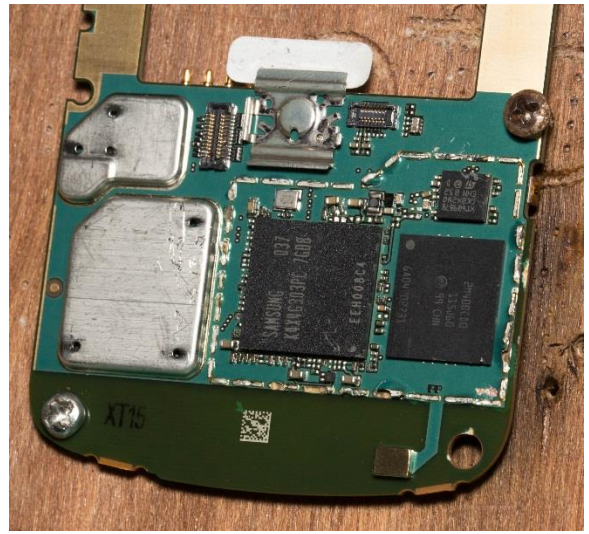
```
allow untrusted_app media_rw_data_file:dir create_dir_perms;
allow untrusted_app media_rw_data_file:file create_file_perms;
```

SEAndroid Danger

Let's relabel `/data/security` from `security_file` to `media_rw_data_file` using an off-line attack or root exploit!

`external/libselinux/src/android.c`

```
static const char *const sepolicy_file[] = {
        "/sepolicy",
        "/data/security/current/sepolicy",
        NULL
};
```



Pathtrust on the Priv

Prevent applications installed on userdata from running with super-user permissions.

https://github.com/blackberry/android-linux-kernel/tree/msm8992/AAF153/security/pathtrust

What is Super-User?

- Specific capabilities
  - `CAP_DAC_OVERRIDE, CAP_MAC_ADMIN, CAP_SYS_ADMIN, CAP_MODULE, ...`
- Specific SEAndroid types
  - `u:r:init:s0, u:r:kernel:s0, u:r:vold:s0, u:r:tee:s0, ...`

What is “Run”?

- Executed through execve
- Mmap’d with exec permissions
- Loaded as a kernel module
- Loaded as firmware
- Executed as a script

Running Binaries

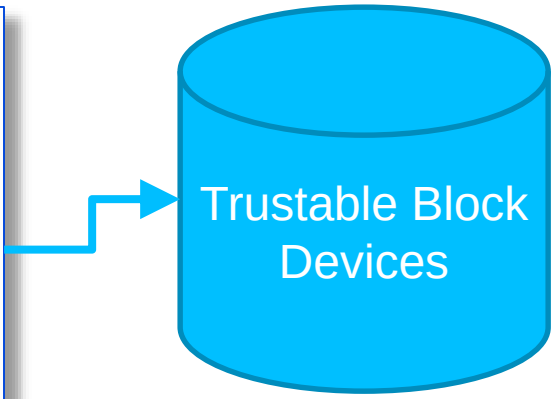
```
root@Priv:/ # cd /data/local/tmp
root@Priv:/data/local/tmp # cp /system/bin/sh .
root@Priv:/data/local/tmp # ./sh
/system/bin/sh: ./sh: Operation not permitted
```

Running Scripts

```
root@Priv:/ # cd /data/local/tmp
root@Priv:/data/local/tmp # cat <<EOF >pathtrust.sh
> #!/system/bin/sh
>
> echo “Hi there”
> EOF
root@Priv:/data/local/tmp # sh pathtrust.sh
sh: pathtrust.sh: Operation not permitted
```

How Does it Work?

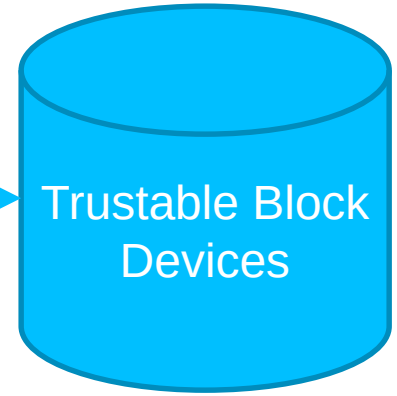
```
static int verity_ctr(struct dm_target *ti, ...) {
   /* Do standard verity checks */
   ...

   /* Add device to trusted block devices */
   pathtrust_add_dev( block_device->bd_dev );
   return 0;
}
```



1. After verifying a block device in dm-verity, add it to a list of trustable block devices.

How Does it Work?

2. Introduce a new “trusted” mount flag
3. When mounting a system with the trusted flag, verify the block device is trustable in the LSM.




```
mount -t ext4 -o trusted,... /dev/.../system /system
```

```
static int security_sb_mount(...) {
   if (flags & MS_TRUSTED) {
      if( !pathtrust_dev_trusted(block_device->bd_dev)) {
         return -EPERM;
      }
   }
   return 0;
}
```

How Does it Work?

4. In the LSM, intercept calls to `mmap`, `bprm_set_creds`, ...
5. If we're trying to run with super-user permissions, verify underlying file-system is trusted.

```
static int security_mmap_file( struct file *file, ..., unsigned long flags ) {
   if (file && ((prot & PROT_EXEC) ||
      ((prot & PROT_READ) && (current->personality & READ_IMPLIES_EXEC))))
   {
      if( !(file->f_path.mnt->mnt_flags & MNT_TRUSTED) &&
         (is_root(cred) || has_banned_caps() || is_forbidden_sid(current->sid))
      {
         return -EPERM;
      }
   }
   return 0;
}
```

How Does it Work?

6. Export a device node for script interpreters to query whether the file should be trusted.

```
static long pathtrust_ioctl ( struct file *fp, unsigned cmd,
                              unsigned long value ) {
   ...
   case IOCTL_TRUST_FILE:
      struct file *file = fget(value);

      if( !(file->f_path.mnt->mnt_flags & MNT_TRUSTED) &&
         (is_root(cred) || has_banned_caps() || is_forbidden_sid(current->sid))
      {
         return -EPERM;
      }
      return 0;
   ...
}
```

Pitfalls

Focused on Preventing Code From Running

- Does not stop a privileged application from accessing untrusted files.
- Only stops a privileged application from invoking attacker-provided code in a separate binary

Isn’t Access Control what SELinux is for?

SELinux already does fine-grained access control, but it doesn’t know about integrity.

Per-File Encryption support for EXT4

Are the xattr records integrity protected in per-file encryption for Android?

Not currently★

★ Possible future research direction

Discussion / Debate

Debate

- Should we include the notion of integrity into SELinux?
- Should integrity protection require encryption?
- What is the relative priority of protecting metadata?
- Can we do it generically with a loop block driver and dm-crypt?

Debate

- Should we include the notion of integrity into SELinux?
  - New type classes?
    - `allow init config_file:trusted_file { open, read, getattr };`
  - New permissions on already-existing type classes?
    - `allow init config_file:file { open, read_trusted, getattr };`
  - Restrict labels that can be applied to file-systems which are not integrity protected?
    - `trusted_label config_file;`

Debate

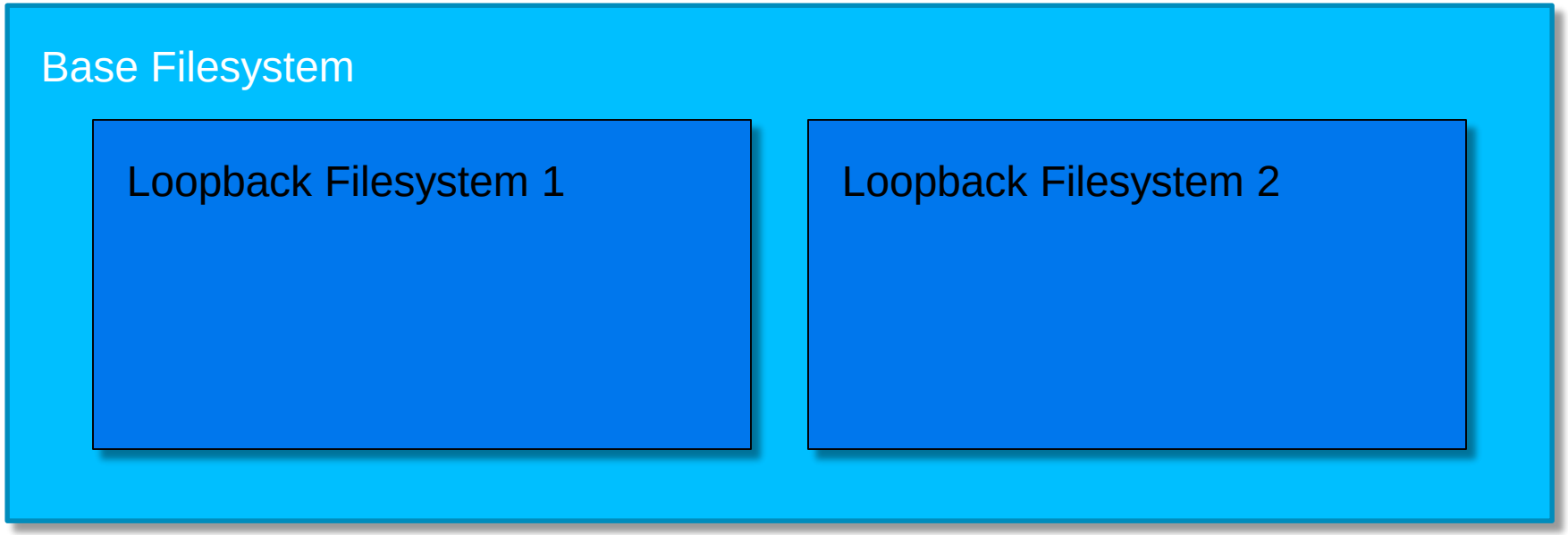
- Should integrity protection require encryption?
  - Is there a use in doing one without the other?
  - Do we encourage insecurity by allowing one without the other?

Debate

- What is the relative priority of protecting metadata?
  - Is protecting a file name more important than protecting the permissions?
  - What's the most common threat model?
  - File-systems are complex, what is the chance that we miss something?

Debate

- Can we do it generically with a loop block driver and dm-crypt?
  - Create a sparse file the size of the base filesystem.
  - Initialize the sparse file with dm-crypt and create a file-system on it.
  - Enable hole-punching when mounting the sparse file-system
  - Free space isn't reported correctly ☹



Thanks!

Glenn Wurster (gwurster@blackberry.com)